18312911011

Ashutosh Jha

Assignment -1

Write a Matlab Code for Bisection Method and Newton Method. Compare these two methods for function

f(x)=3x + sinx - ex

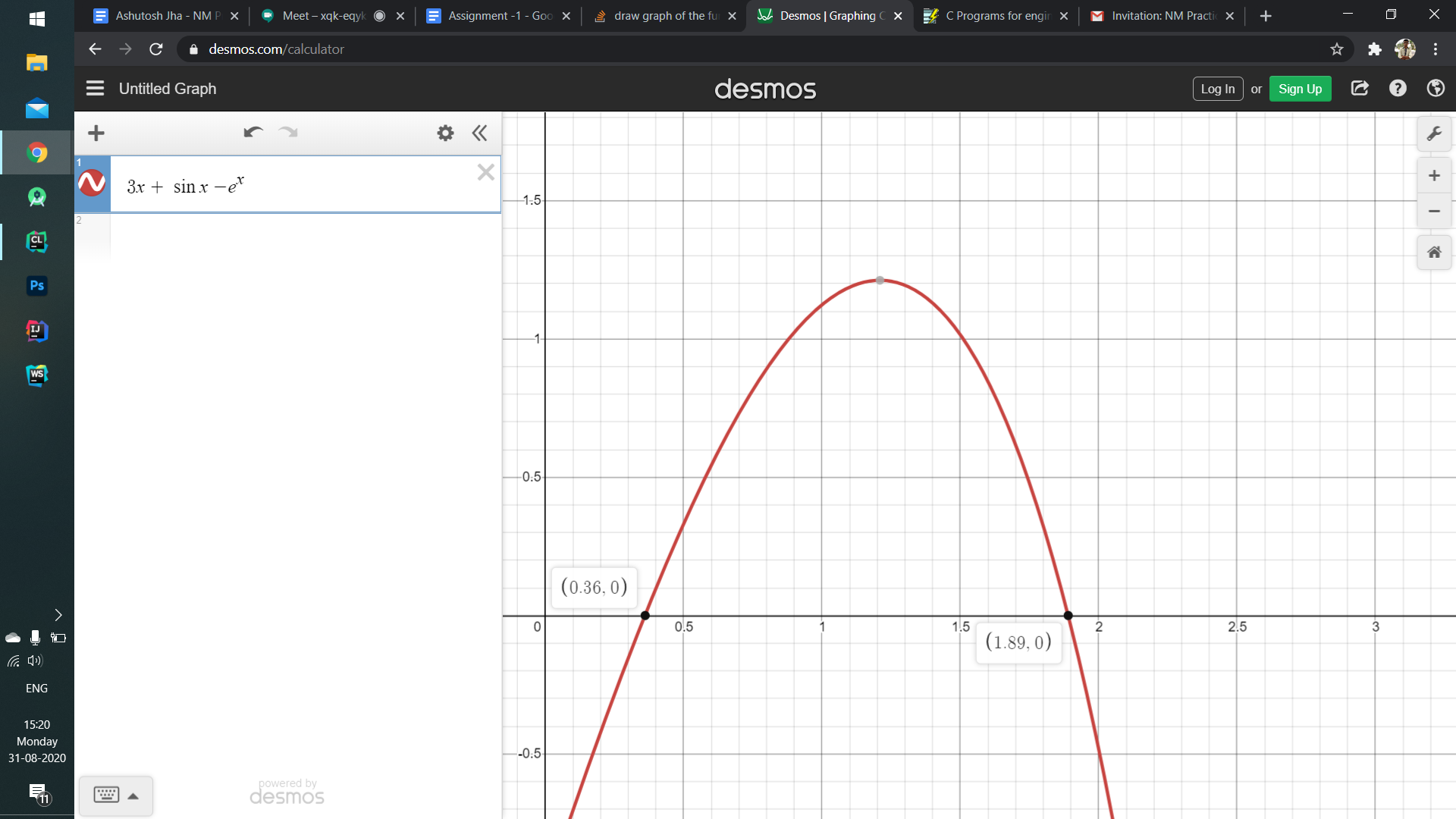
Note: For Newton’s Method initial guess is 0.

Plot your solution for these methods and write a note on your observation regarding convergence of these methods.

**Bisection Method :**

Plot of f(x)=3x + sinx - ex

(Plotted using Desmos)



So, In bisection method we take two intervals, a and b for each root

such that

f(a) \* f(b) < 0

For finding first root in interval (1, 2)

No of iterations done : 15

r = 1.89003 (Found using calculator for finding error)

a = 1, b= 2

Program

(Done in C)

*// Bisection Method*

#include<bits/stdc++.h>

*// Maximum permissible error*

#define **MAX\_ERROR** 0.000001

*//No. of iterations done*

int iterations = 0;

double rootValues[20];

double actualError[20];

double maximumError[20];

double functionValueForC[20];

*//Function given in question*

double func(double x)

{

return 3\*x - exp(x) + sin(x);

}

double answer() {

*//Actual root found using calculator;*

return 1.89003;

}

*//Function to find the root using bisection Method*

void bisectionMethod(double x\_lower, double x\_mid, double x\_upper)

{

while (abs(func(x\_mid)) >= **MAX\_ERROR**)

{

if(func(x\_mid) \* func(x\_upper) < 0) {

x\_lower = x\_mid;

} else if (func(x\_mid) \* func(x\_lower) < 0){

x\_upper = x\_mid;

} else {

break;

}

rootValues[iterations] = x\_mid;

functionValueForC[iterations] = func(x\_mid);

actualError[iterations] = x\_mid - answer();

maximumError[iterations] = (x\_upper - x\_lower) / 2;

iterations = iterations + 1;

x\_mid = (x\_upper+x\_lower) / 2;

}

*//Printing the root*

std::cout << "The value of the root is : " << x\_mid;

*//No. of iterations done for maximum error*

std::cout << "\nThe value of the iterations is : " << iterations;

*//Printing the x\_mid for each iteration*

std::cout << "\nx\_mid values";

for(double rootValue : rootValues) {

std::cout << "\n" << rootValue;

}

*//Printing the f(c) value*

std::cout << "\nf(x\_mid)";

for (double i : functionValueForC){

std::cout << "\n" << i;

}

*//Printing the actual error*

std::cout << "\nActual error";

for (double i : actualError){

std::cout << "\n" << i;

}

*//Printing maximum error*

std::cout << "\nMaximum Error";

for (double i : maximumError){

std::cout << "\n" << i;

}

}

int main()

{

double x\_lower = 1;

double x\_upper = 2;

double x\_mid = (x\_lower + x\_upper) / 2;

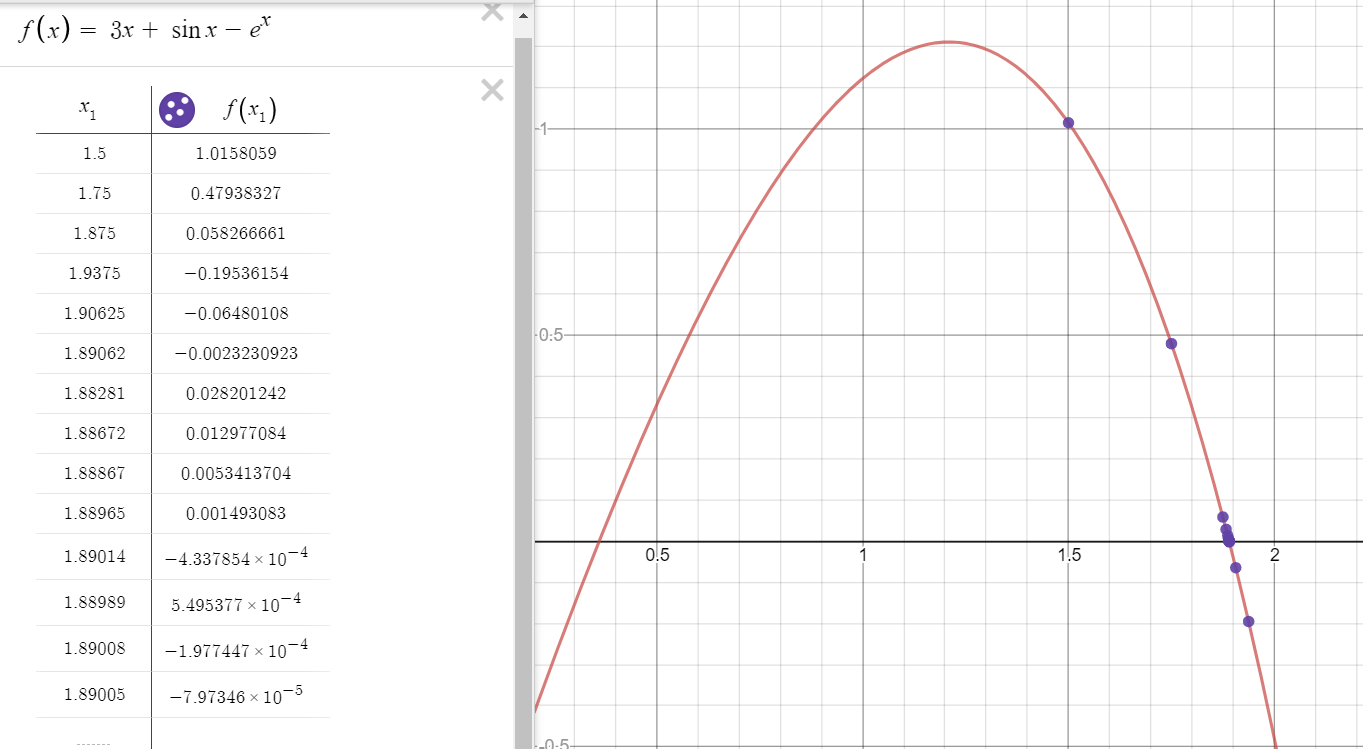
*// Initial values assumed*

bisectionMethod(x\_lower, x\_mid, x\_upper);

return 0;

}

| Iterations | c = (a + b) / 2 | f(c) | Actual Error |
| --- | --- | --- | --- |
|  |  |  |  |
| 1 | 1.5 | 1.01581 | -0.39003 |
| 2 | 1.75 | 0.479383 | -0.14003 |
| 3 | 1.875 | 0.0582667 | -0.01503 |
| 4 | 1.9375 | -0.195362 | 0.04747 |
| 5 | 1.90625 | -0.0648011 | 0.01622 |
| 6 | 1.89062 | -0.00234278 | 0.000595 |
| 7 | 1.88281 | 0.0281915 | -0.0072175 |
| 8 | 1.88672 | 0.012982 | -0.00331125 |
| 9 | 1.88867 | 0.00533401 | -0.00135813 |
| 10 | 1.88965 | 0.00149922 | -0.000381563 |
| 11 | 1.89014 | -0.000420876 | 0.000106719 |
| 12 | 1.88989 | 0.0005394 | -0.000137422 |
| 13 | 1.89001 | 5.93E-05 | -1.54E-05 |
| 14 | 1.89008 | -0.000180765 | 4.57E-05 |
| 15 | 1.89005 | -6.07E-05 | 1.52E-05 |



For finding first root in interval (0, 1)

No of iterations done : 20

r = 0.36042170 (Found using calculator for finding error)

a = 0, b= 1

Program

(Done in C)

*// Bisection Method*

#include<bits/stdc++.h>

*// Maximum permissible error*

#define **MAX\_ERROR** 0.000001

*//No. of iterations done*

int iterations = 0;

double rootValues[20];

double actualError[20];

double maximumError[20];

double functionValueForC[20];

*//Function given in question*

double func(double x)

{

return 3\*x - exp(x) + sin(x);

}

double answer() {

*//Actual root found using calculator;*

return 0.36042170;

}

*//Function to find the root using bisection Method*

void bisectionMethod(double x\_lower, double x\_mid, double x\_upper)

{

while (abs(func(x\_mid)) >= **MAX\_ERROR**)

{

if(func(x\_mid) \* func(x\_upper) < 0) {

x\_lower = x\_mid;

} else if (func(x\_mid) \* func(x\_lower) < 0){

x\_upper = x\_mid;

} else {

break;

}

rootValues[iterations] = x\_mid;

functionValueForC[iterations] = func(x\_mid);

actualError[iterations] = x\_mid - answer();

maximumError[iterations] = (x\_upper - x\_lower) / 2;

iterations = iterations + 1;

x\_mid = (x\_upper+x\_lower) / 2;

}

*//Printing the root*

std::cout << "The value of the root is : " << x\_mid;

*//No. of iterations done for maximum error*

std::cout << "\nThe value of the iterations is : " << iterations;

*//Printing the x\_mid for each iteration*

std::cout << "\nx\_mid values";

for(double rootValue : rootValues) {

std::cout << "\n" << rootValue;

}

*//Printing the f(c) value*

std::cout << "\nf(x\_mid)";

for (double i : functionValueForC){

std::cout << "\n" << i;

}

*//Printing the actual error*

std::cout << "\nActual error";

for (double i : actualError){

std::cout << "\n" << i;

}

*//Printing maximum error*

std::cout << "\nMaximum Error";

for (double i : maximumError){

std::cout << "\n" << i;

}

}

int main()

{

double x\_lower = 0;

double x\_upper = 1;

double x\_mid = (x\_lower + x\_upper) / 2;

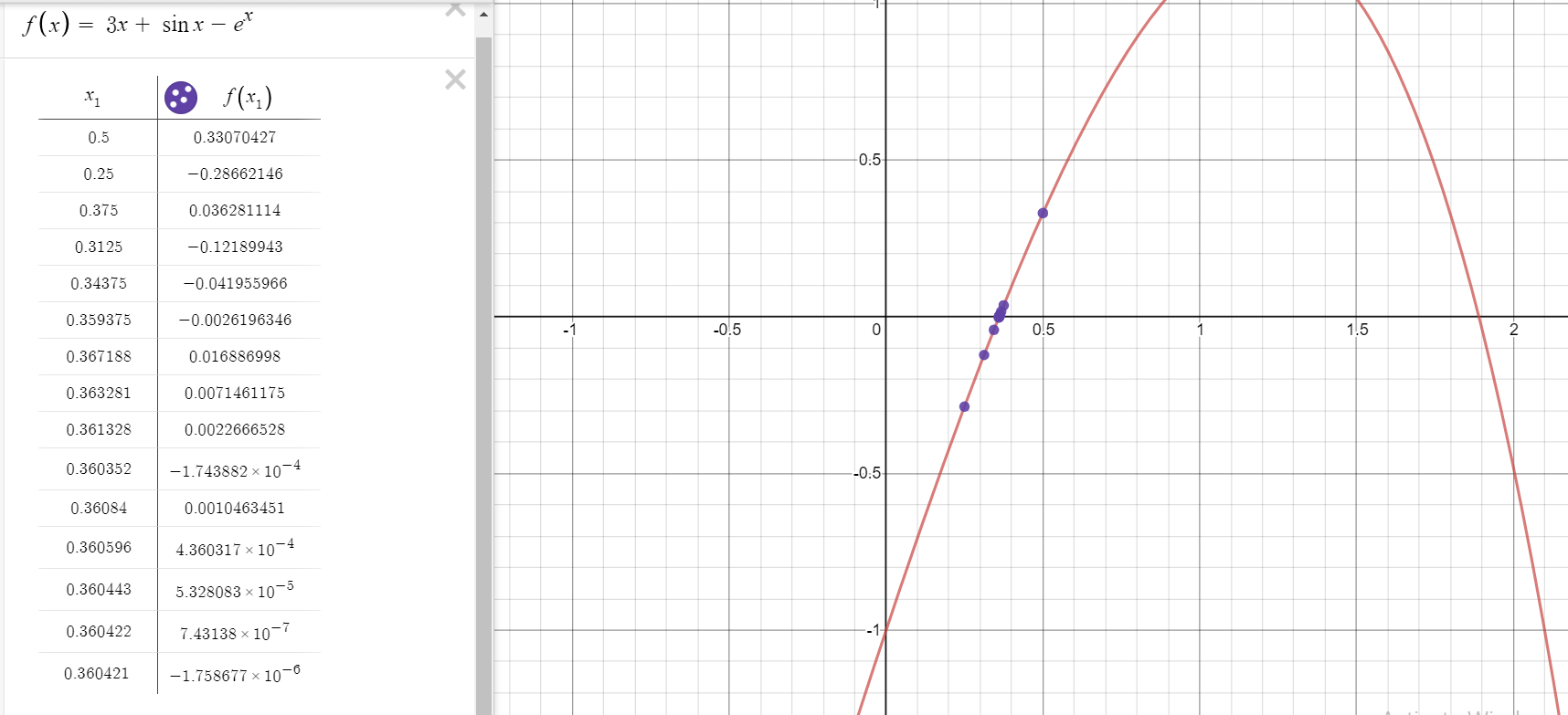
*// Initial values assumed*

bisectionMethod(x\_lower, x\_mid, x\_upper);

return 0;

}

| Iterations | c = (a + b) / 2 | f(c) | Actual Error |
| --- | --- | --- | --- |
|  |  |  |  |
| 1 | 0.5 | 0.330704 | 0.139578 |
| 2 | 0.25 | -0.286621 | -0.110422 |
| 3 | 0.375 | 0.0362811 | 0.0145783 |
| 4 | 0.3125 | -0.121899 | -0.0479217 |
| 5 | 0.34375 | -0.041956 | -0.0166717 |
| 6 | 0.359375 | -0.00261963 | -0.0010467 |
| 7 | 0.367188 | 0.0168858 | 0.0067658 |
| 8 | 0.363281 | 0.00714674 | 0.00285955 |
| 9 | 0.361328 | 0.00226697 | 0.000906425 |
| 10 | 0.360352 | -0.000175483 | -7.01E-05 |
| 11 | 0.36084 | 0.00104595 | 0.000418144 |
| 12 | 0.360596 | 0.000435289 | 0.000174003 |
| 13 | 0.360474 | 0.000129916 | 5.19E-05 |
| 14 | 0.360413 | -2.28E-05 | -9.10E-06 |
| 15 | 0.360443 | 5.36E-05 | 2.14E-05 |
| 16 | 0.360428 | 1.54E-05 | 6.16E-06 |
| 17 | 0.36042 | -3.69E-06 | -1.47E-06 |
| 18 | 0.360424 | 5.85E-06 | 2.34E-06 |
| 19 | 0.360422 | 1.08E-06 | 4.34E-07 |
| 20 | 0.360421 | -1.31E-06 | -5.19E-07 |



**Newton’s Method**

For finding first root in interval (1, 2)

No of iterations done : 5

r = 1.89003 (Found using calculator for finding error)

x0 = 2;

Program

(Done in C)

#include<bits/stdc++.h>

#define **MAX\_ERROR** 0.0000000001

using namespace std;

int iterations = 0;

double values[10];

double actualError[20];

double functionValueForC[20];

double func(double x)

{

return 3\*x - exp(x) + sin(x);

}

double answer() {

*//Actual root found using calculator;*

return 1.89003;

}

double derivFunc(double x)

{

return 3 - exp(x) + cos(x);

}

*// Function to find the root*

void newtonRaphson(double x)

{

double h = func(x) / derivFunc(x);

while (abs(h) >= **MAX\_ERROR**)

{

h = func(x)/derivFunc(x);

*// x(i+1) = x(i) - f(x) / f'(x)*

x = x - h;

values[iterations] = x;

iterations = iterations + 1;

functionValueForC[iterations] = func(x);

actualError[iterations] = x - answer();

}

cout << "The value of the root is : " << x; *//Printing the x\_mid for each iteration*

*//Printing the x\_mid for each iteration*

std::cout << "\nx\_mid values";

for(double midValue : values) {

std::cout << "\n" << midValue;

}

*//Printing the f(c) value*

std::cout << "\nf(x\_mid)";

for (double i : functionValueForC){

std::cout << "\n" << i;

}

*//Printing the actual error*

std::cout << "\nActual error";

for (double i : actualError){

std::cout << "\n" << i;

}

}

int main()

{

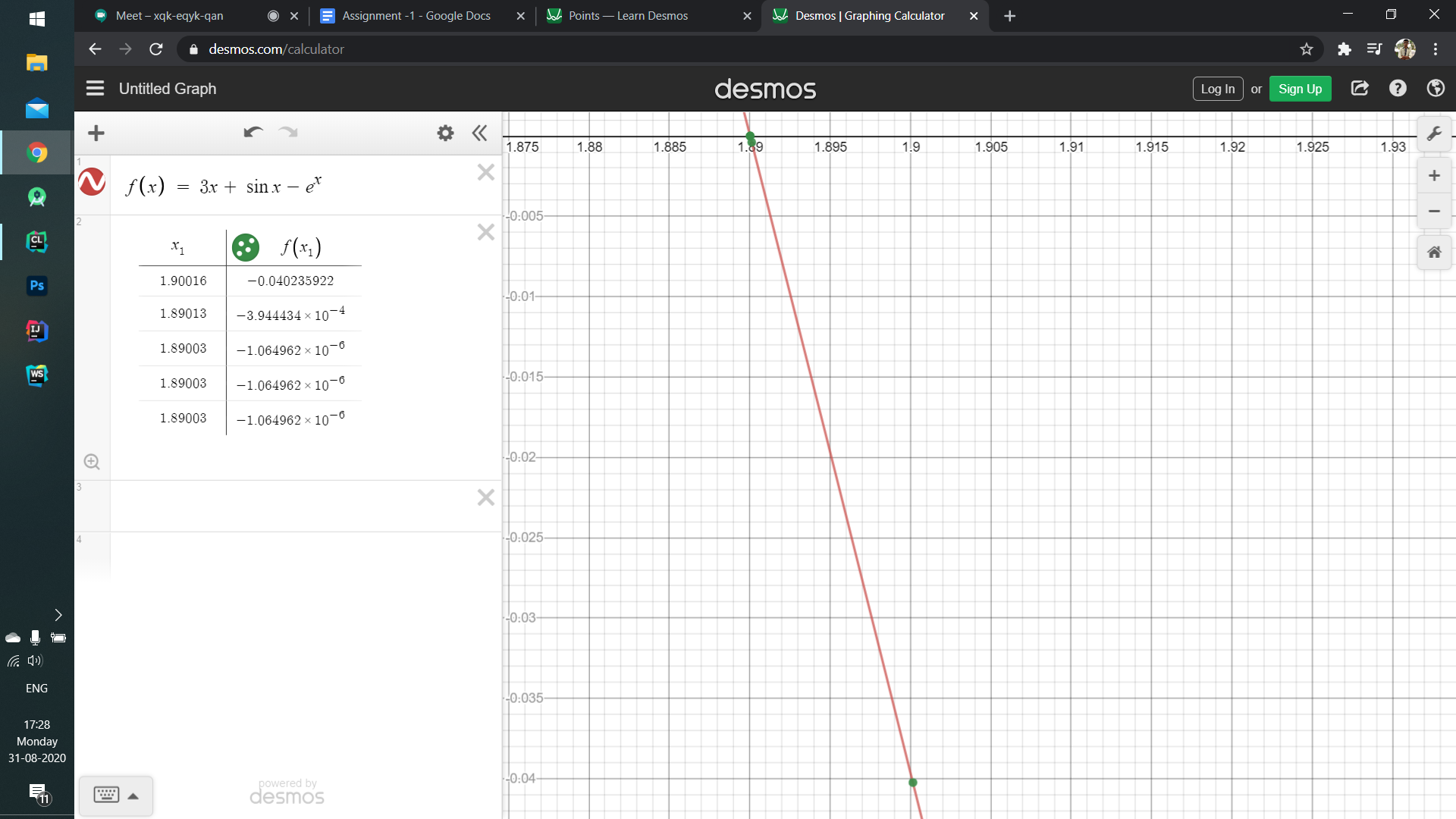
double x0 = 2; *// Initial value which is given to 0*

newtonRaphson(x0);

return 0;

}

| Iterations | c = (a + b) / 2 | f(c) | Actual Error |
| --- | --- | --- | --- |
|  |  |  |  |
| 1 | 1.90016 | -0.0402299 | 0.0101285 |
| 2 | 1.89013 | -0.000382993 | 9.71E-05 |
| 3 | 1.89003 | -3.59E-08 | -2.62E-07 |
| 4 | 1.89003 | -1.11E-16 | -2.71E-07 |
| 5 | 1.89003 | -1.11E-16 | -2.71E-07 |



For finding first root in interval (1, 2)

No of iterations done : 5

r = 0.36042170 (Found using calculator for finding error)

x0 = 0;

#include<bits/stdc++.h>

#define **MAX\_ERROR** 0.0000000001

using namespace std;

int iterations = 0;

double values[10];

double actualError[20];

double functionValueForC[20];

double func(double x)

{

return 3\*x - exp(x) + sin(x);

}

double answer() {

*//Actual root found using calculator;*

return 0.3604217;

}

double derivFunc(double x)

{

return 3 - exp(x) + cos(x);

}

*// Function to find the root*

void newtonRaphson(double x)

{

double h = func(x) / derivFunc(x);

while (abs(h) >= **MAX\_ERROR**)

{

h = func(x)/derivFunc(x);

*// x(i+1) = x(i) - f(x) / f'(x)*

x = x - h;

values[iterations] = x;

iterations = iterations + 1;

functionValueForC[iterations] = func(x);

actualError[iterations] = x - answer();

}

cout << "The value of the root is : " << x; *//Printing the x\_mid for each iteration*

*//Printing the x\_mid for each iteration*

std::cout << "\nx\_mid values";

for(double midValue : values) {

std::cout << "\n" << midValue;

}

*//Printing the f(c) value*

std::cout << "\nf(x\_mid)";

for (double i : functionValueForC){

std::cout << "\n" << i;

}

*//Printing the actual error*

std::cout << "\nActual error";

for (double i : actualError){

std::cout << "\n" << i;

}

}

int main()

{

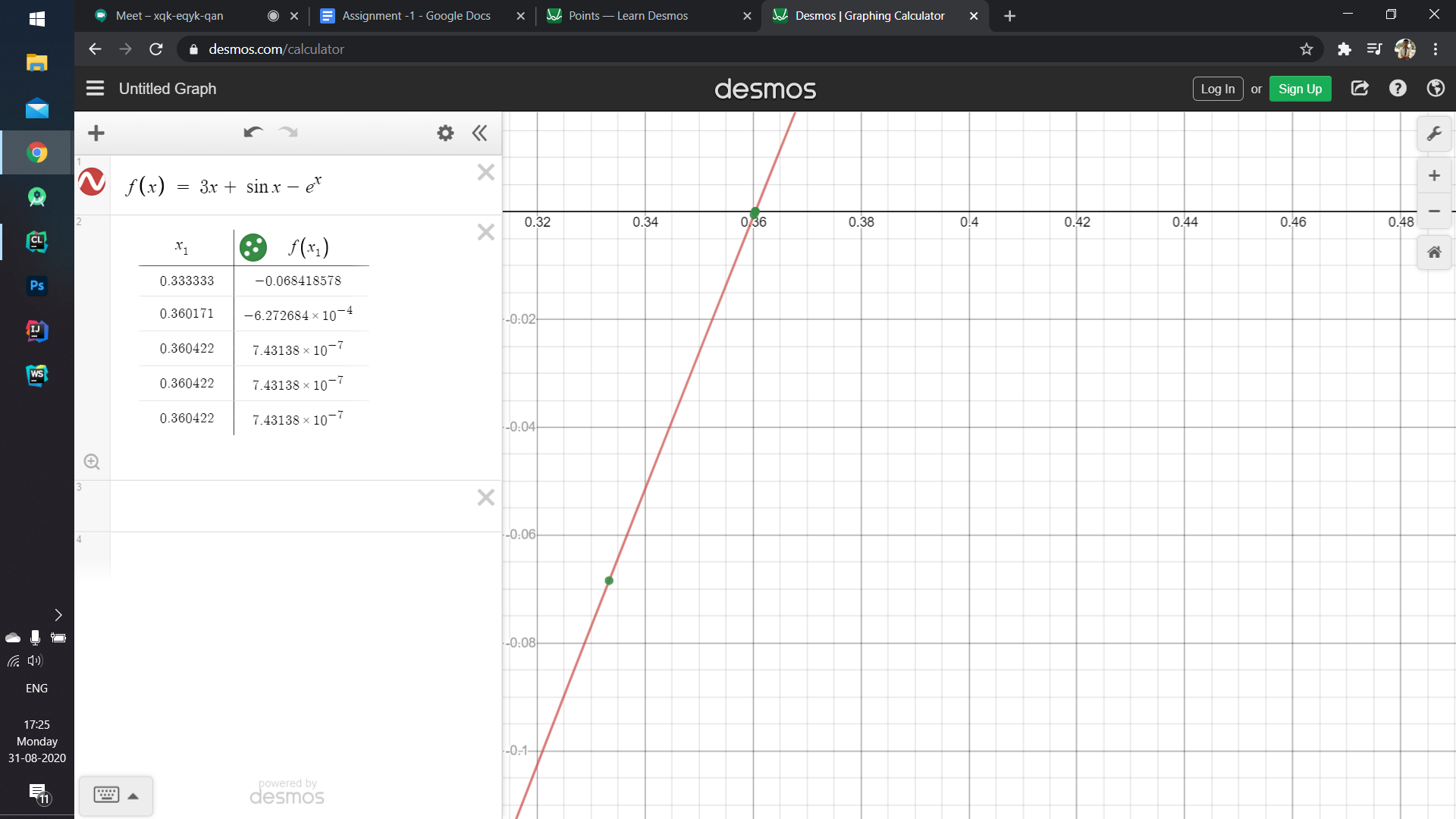
double x0 = 0; *// Initial value which is given to 0*

newtonRaphson(x0);

return 0;

}

| Iterations | c = (a + b) / 2 | f(c) | Actual Error |
| --- | --- | --- | --- |
|  |  |  |  |
| 1 | 0.333333 | -0.0684177 | -0.0270884 |
| 2 | 0.360171 | -0.000627985 | -0.000250986 |
| 3 | 0.360422 | -5.63E-08 | -1.95E-08 |
| 4 | 0.360422 | -5.00E-16 | 2.96E-09 |
| 5 | 0.360422 | 1.11E-16 | 2.96E-09 |



**Convergence Analysis**

Bisection Method - It is converging in this case as the actual error keeps on decreasing till zero and the value of c is approaching towards the root i.e. 0.360422 and 1.89003. We needed to do 15-20 iterations for allowing a maximum error of 0.000001

Newton Method - It is also converging in this case as the actual error keeps on decreasing till zero and the value of c is approaching towards the root i.e. 0.360422 and 1.8900. We needed to do 4-5 iterations for allowing a maximum error of 0.0000000001

**Conclusion**

Newton method is fast and more optimized as we needed only 5 iterations for allowing a max error of 0.0000000001 whereas in Bisection we needed 15-20 iterations for allowing a maximum error of 0.000001.

So, I conclude that newton method is much better than bisection